VOL. 15, NO. 22, NOVEMBER 2020 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

# VIRTUAL ENVIRONMENT FOR SMART ROBOTIC APPLICATIONS

Javier Pinzon-Arenas, Robinson Jimenez-Moreno and Astrid Rubiano
Faculty of Engineering, Universidad Militar Nueva Granada, Bogotá, Colombia
E-Mail: u3900231@unimilitar.edu.co

## ABSTRACT

This article presents the development of a virtual environment for testing industrial-type robotic applications, using artificial intelligence techniques through convolutional neural networks. The developed environment is made up of two RGB-D cameras for capturing information from the workspace in which the robot operates a conveyor belt and a work table. A ResNet-50 model within a Faster R-CNN is used to detect a tool, in order to indicate to the robot the position of it, to generate a pick and place application, from the table to the conveyor belt, as a method for validating the environment, reaching an average precision of detection of the object of interest greater than 82% and an 84% success rate on the pick and place task.

**Keywords:** faster R-CNN, robotic application, virtual environment.

## INTRODUCTION

Technological advances have allowed robotics venture into many applications in the industry (Shanshan He, 2020) (Hsien-I Lin), being the simulated environments one of the main tools for the development of algorithms that allow evaluating the functionality of a robotic system (Yue Wang, 2020) (Qirong Tang), which, before its physical implementation, opens the possibility to propose multiple solutions and ensure the conditions for a real testing environment (Vladimir Kvrgic).

Nowadays, one of the techniques that is integrating automation task algorithms for robotic agents is the convolutional neural network (CNN) (Arenas, 2018). Many autonomous systems use CNNs for pattern recognition given their versatility, for instance, cardiac image analysis for localization of the heart, as presented in (Saeed Kermani). Other example is vehicle detection by using CNN variants such as Faster R-CNN that, by means of region proposals, determines the location of an object in an image or video sequence (Saeed Kermani) (Wei Liu). Robotic applications based on the Faster R-CNN exhibit various case studies, from precision agriculture applications (Shaohua Wan), e.g. strawberry detection and harvesting (Yang Yu), to the location of objects for ordering tools (Arenas, 2018).

Application cases such as presented in (Muhammad) focus on the use of region-based CNNs in industrial environments for object recognition and tracking in manufacturing tasks. Those aspects highlight the importance and functionality of integrating these techniques in the automation of robotic tasks, where simulated environments present a significant tool of analysis and development.

Derived from the arguments exposed previously, this article presents the implementation of a virtual environment for simulate operations with robots using artificial intelligence techniques such as convolutional networks based on regions. According to (Yueyue Liu), the application of Learning Transfer techniques allows the use of CNN architectures presented in the state of the art (Qian Zhang) in robotic tasks based on RGB-D information. Thus, the proposed environment uses RGB-D information, the use of region-based CNN through the transfer of learning for object recognition and location, as well as a robotic arm capable of performing industrial-type tasks.

The environment is validated using a pick and place application in an industrial environment, where the system detects and indicates to the robot the location of it, in order to be grasped and placed on a conveyor belt. The environment is open to collaborative robotics works and even more to man-machine interaction works which have also gained great force in the state of the art (Emanuele) (Federica) (Seyyedhasani).

The article is structured as follows: the present section corresponds to the introduction and review of the state of the art. Section 2 presents the methodology used and the training of the CNN-type network. Section 3 presents the analysis and results of the virtual environment in the pick and place application and ends with section 4, in which the conclusions reached are presented.

## METHODS AND MATERIALS

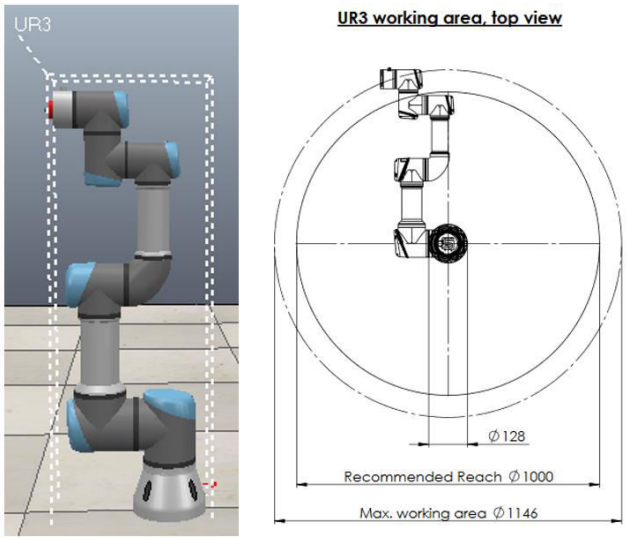
For the elaboration of the virtual environment, the software CoppeliaSim was used, which allows an interaction between the simulation and different programming software, to control different elements or devices to be used within the virtual environment, such as vision sensors, e.g. RGB and depth cameras; force and torque sensors, for when it is required to know the dynamic parameters of a robotic element; among others.

CoppeliaSim also allows setting and adjusting parameters related to the physics and dynamics of the objects existing in the environment, which is an important factor to perform a simulation close to reality, so that, in the case of the environment to be implemented, the objects can be manipulated by the robot, transported on conveyors, stored, etc., depending on the pick and place application to be developed.

On the other hand, a great advantage of this software is that it has licenses available for different robotic models, among which is the UR3 robotic manipulator (Figure-1a) that will be used in different applications within the project. Additionally, it allows

simulating its kinematics, thus facilitating its control and operation to execute different tasks.

For the construction of the virtual environment, the following elements are available: an UR3 robotic device, 2 RGB-D cameras, 5 objects of interest, and a conveyor belt.




(a) (b)

**Figure-1.** a) UR3 Robot. b) Maximum lateral range of the robot.

The robot is placed on a work table with dimensions of 1.2x1.2 m wide and long, and a height of 0.9 m high, in which it maintains a space on each side for tool relocation and arrangement, if necessary (Figure-2). On the table, the robot is located 25 cm from the nearest edge to the center of its base and centered 60 cm to the side and side of the two lateral edges, which allows taking advantage of the lateral spaces, since the maximum reach of the robot is 57.3 cm (Figure-1b). As for its front part, a space of 95 cm is maintained, with an available space of 37.7 cm, to locate the tools that do not need to be ordered, or are in line to be placed by the user in front of the work table.

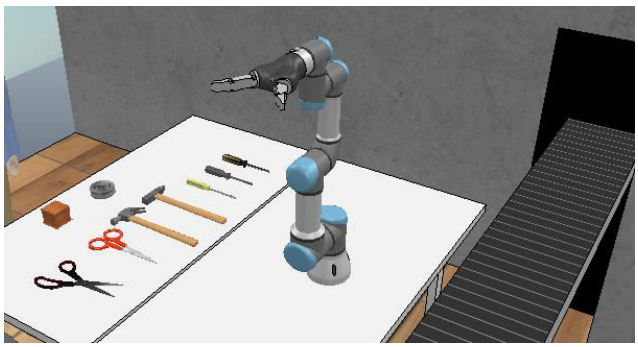


**Figure-2.** Location of the robot in the virtual environment.

A conveyor belt is placed, which can bring or carry the different objects as required in the application. Its location can be seen in Figure-2. The current parameters of the conveyor are a length of 2 meters, a width of 30 centimeters and a height of 90 centimeters. Also, its speed is 0.1 m/s (it can reach a maximum speed of 1 m/s) in the direction of the robot towards the tunnel.

Among the objects of interest are scissors (two types), hammers (two types), screwdrivers (three types), a pulley, and a wooden box. Each object can be seen in Figure-3.

**Figure-3.** Objects placed to be manipulated.

To add the depth cameras to the environment, these must be parameterized according to the characteristics of the camera to be used. In this case, it is configured with a viewing angle of 86°, with a minimum operating range of 10 cm and a maximum of 2 m, according to the characteristics of an Intel RealSense D435i camera. Likewise, a resolution of 640x480 pixels is set, since for applications with deep learning techniques, especially focused on object detection, this resolution is sufficient. Using higher resolutions can significantly increase the application's processing time.

In order to take the images of what the robot sees from its end effector, a local camera is available on it, as shown in Figure-4 (upper). To verify that it can see the objects from a vertical positioning of the robot, the test is performed by rotating one of its joints in the direction of the tools, obtaining the image shown in Figure-4 (lower). The figure shows how the robot sees the objects from its perspective, as well as the depth map, in which the objects are displayed on the table, calibrated for a maximum depth of 1 meter, since the robot does not require recognition at a greater distance since its maximum range is this distance. In case it is required to generate a larger depth map, the parameters can be modified as explained above.

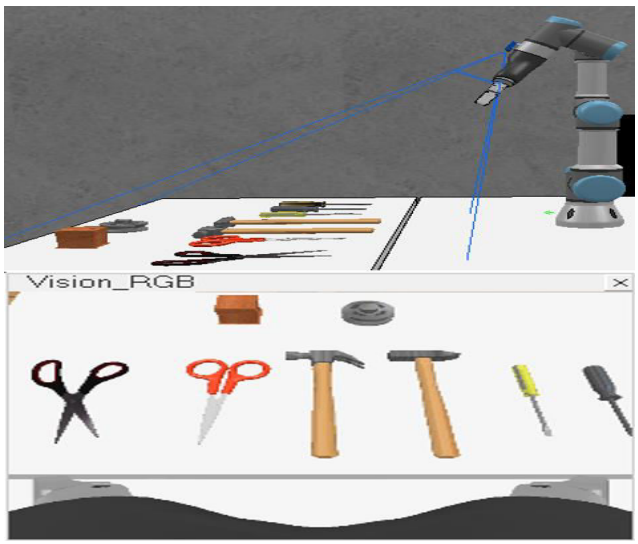



**Figure-4.** Global camera localization.

The use of two cameras, one fixed and one mobile, allows versatility in the range of applications that can be simulated within the environment, so that if the perspective of the robot is mainly required, the mobile camera arranged in the end effector can be used. However, if a camera is required to observe the environment in a general way, and remain fixed in one position, the superior global camera may be used. There may even be applications where both cameras are necessary.

**Test application**

To verify the functionality of the virtual environment, a pick and place application is made. In this application, an object of interest will be recognized and located by means of the global camera. In this case, the object will be a screwdriver, which will be detected by a neural network based on Deep Learning. After being detected, the robot will collect the object and place it on the conveyor belt. Eventually, the robot will return to its initial position. In this case, the mobile camera of the manipulator is not used, since the application to be carried out does not require it.

First, the database to be used is created, for which the global camera will be used. With this, images of the environment are acquired by placing the screwdriver in different positions and orientations on the work table at random. Then, each image is manually labeled, where the screwdriver is enclosed in a bounding box, which indicates the location of the object within the image. With this procedure, a total of 800 images are obtained for training and 200 for validation of the neural network, with a size of 640x480 pixels each image. An example of the images is shown in Figure-5. Inside the figure, the screwdriver is enclosed in a red box corresponding to the established bounding box.

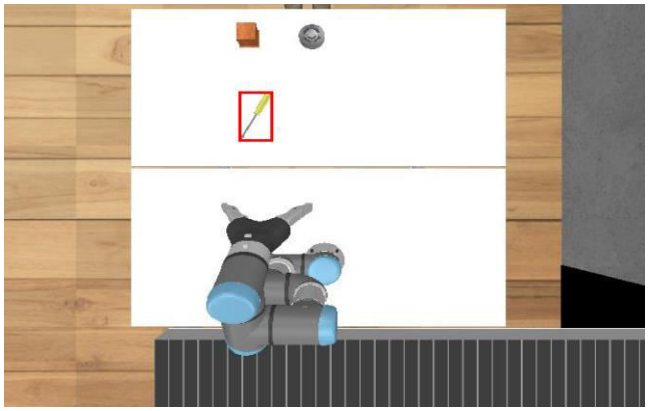


**Figure-5.** Sample of the database built.

Once the database is built, the neural network to be used is established. For this application, it is proposed to use a Faster R-CNN (Ren Shaoqing), which has the capability to detect objects of interest with a high degree of accuracy and speed. In general terms, this neural network is composed of a convolutional neural network (CNN), which before reaching its output, is divided into two paths, one directly to a downsampling layer, and the other entering a region propose network (RPN), in charge of learning the possible locations and shapes of the bounding boxes, to finally join to the downsampling layer. Finally, the linked paths come out into a classification layer, to decide whether or not the proposed regions belong to an object of interest and what the coordinates of this object are. This structure is shown in Figure-6.

As the main structure of the CNN, the ResNet-50 model (He Kaiming) is used, which is composed of 50 convolution layers.

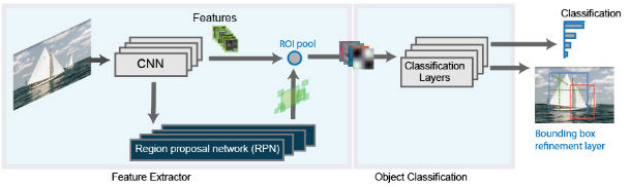



**Figure-6.** Structure of the Faster R-CNN. Source: Mathworks®.

The training of the Faster R-CNN is done in 4 stages. The first stage is in charge of training the section of the RPN independently, to have as output a regression from which the coordinates of the bounding boxes are obtained. The second stage trains the classification section of the network, taking into account the regions proposed in the first stage. The third one trains again the RPN, but in a fine way, that is, varying in a minor factor the weights learned in the previous stages, sharing, in the same way, the weights of the classification part. Finally, in stage four, it is performed a fine-tuning of the classification layers.

Since the architecture is a pre-trained model, it is generally fine-tuned, so its learning rate is low (less than 1x10-3) from the first stage. The number of images used per iteration is 1, since this type of network is trained

image by image. The training parameters at each stage are show in Table-1.

**Table-1.** Training parameters of the Faster R-CNN.

| | Learning Rate | Epochs |
|---|---|---|
| Stage 1 | $1x10^{-3}$ | 10 |
| Stage 2 | $1x10^{-3}$ | 10 |
| Stage 3 | $5x10^{-4}$ | 8 |
| Stage 4 | $5x10^{-4}$ | 8 |

As a result of the network training, it is obtained three types of graphs that show the behavior throughout the training during the four stages. The first one, shown in Figure-7, is the network losses, that is, the cost by mistake at the time of generating the bounding boxes or wrongly classifying the image. During the training stages of the RPN, the network obtained a low and decreasing cost, which means that the network effectively learned to correctly locate the object of interest. In the classification stages, it was more difficult for the network to learn what a screwdriver actually was and that it was not. However, its loss was decreasing and less than 0.5, which allows us to infer that the network had a high rate of correct classifications. This can be seen by looking at Figure-8, where their average accuracy at these stages was over 90%. On the other hand, Figure-9 shows the root mean square error (RMSE), which indicates that the bounding boxes generated by the network had so much error with respect to those proposed in the database, showing errors lower than 1, that is, that the bounding boxes generated were within those proposed.

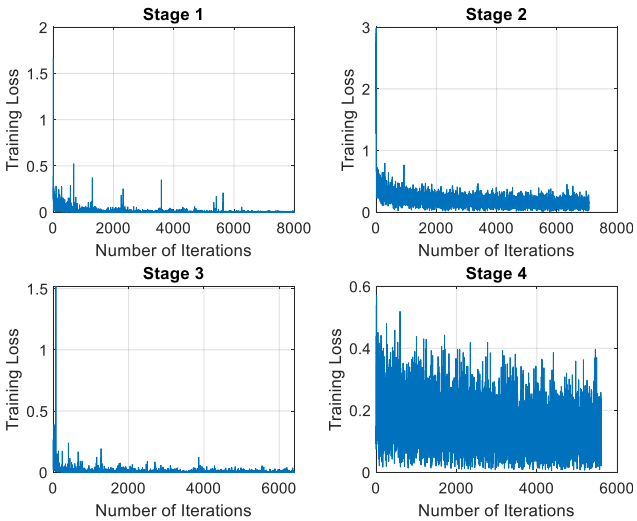



**Figure-7.** Loss at every stage of training.

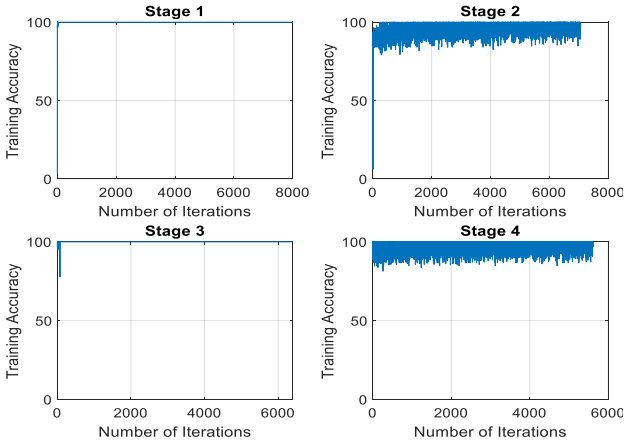



**Figure-8.** Accuracy at every stage of training.

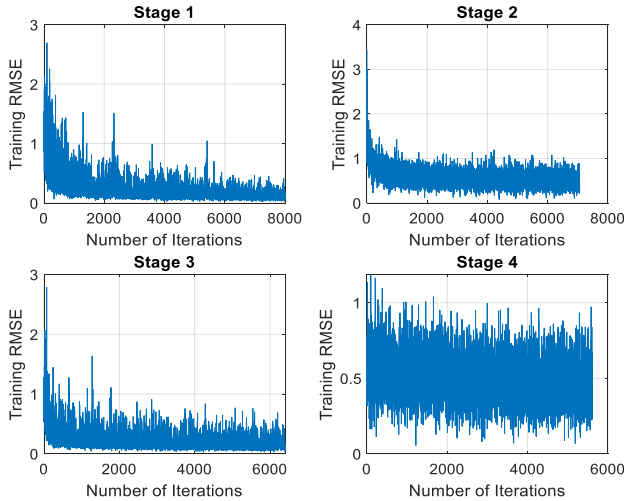



**Figure-9.** Root mean square error at each stage of training.

## RESULTS

First, the network performance tests are carried out. For this, the built validation set is used. An example of network detection for a validation image is shown in Figure-10. As you can see, in upper part, the network actually detects the screwdriver; however, it also locates a wrong box at the top where the pulley is located. When comparing the two detections, it can be seen that the confidence with which it detects the pulley is less than 85%, while for the screwdriver, the network is more than 95% sure that it is the object of interest. For this reason, it was decided to apply a confidence threshold, so that the network would discard objects detected with a threshold of less than 85%. This way, it is obtained the image shown in lower part, where it only detects the screwdriver inside the image.

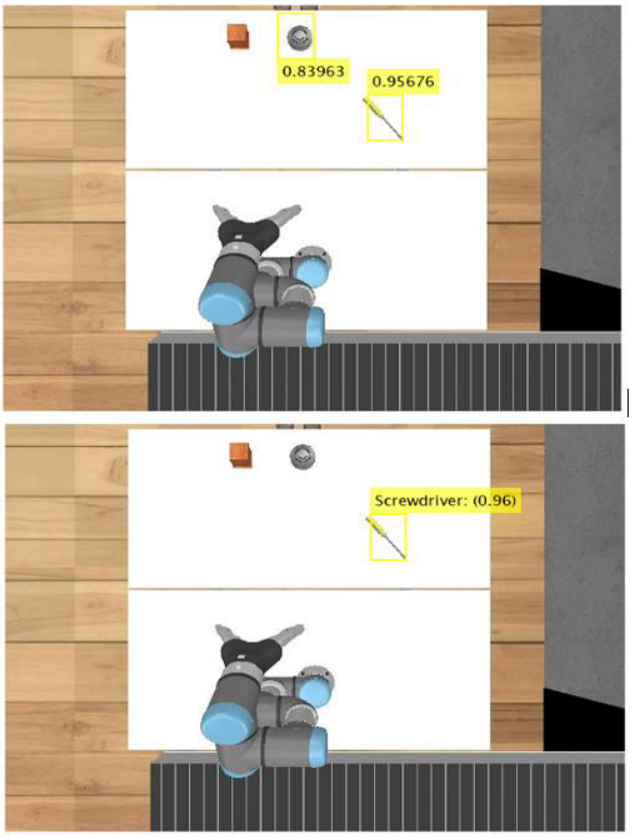



**Figure-10.** Network detection. No confidence threshold (upper). With confidence threshold (lower).

To verify the capacity of the network with the entire validation set, the precision vs recall graph is used. This graph allows to see how well the boxes obtained by the network overlap with those detected manually. This is done with a 0% to 100% overlap evaluation (100% indicates that the box is completely overlapped on the proposal). With this, the graph shown in Figure-11 is obtained, where the network obtains an average accuracy of 82% in terms of screwdriver detection, which is a high value for detection systems (over 75%).

Once the network has been tested, it is coupled to the virtual environment via the global camera. For the application, the RG2 gripper is used, as it allows an easy and firm grip on the objects.

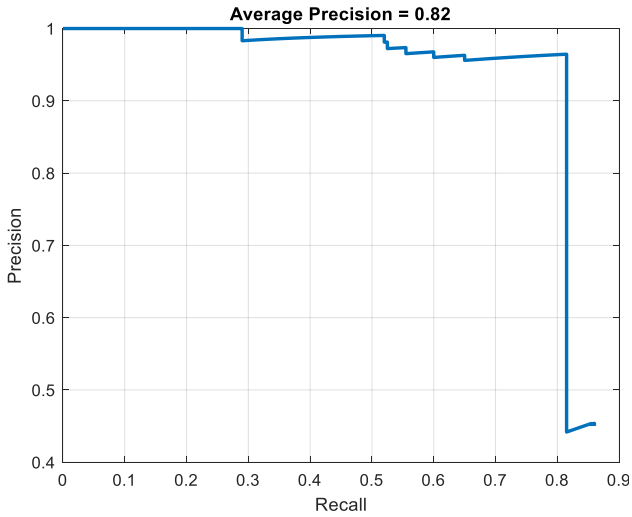
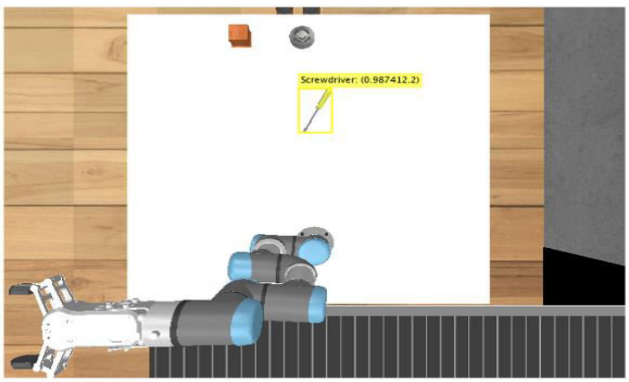
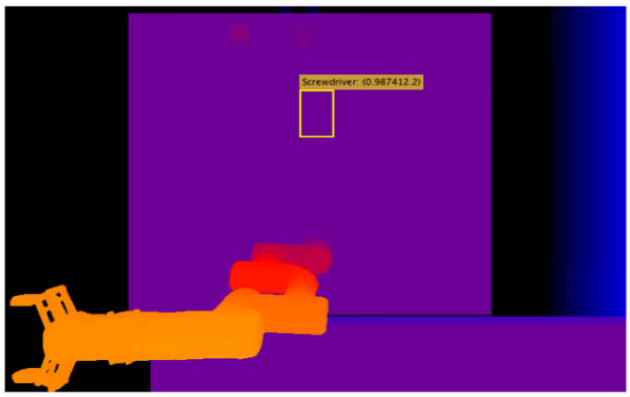



**Figure-11.** Precision vs. Recall of the Faster R-CNN.

The process of the pick and place application begins with the image capture by the global camera, so that this is entered into the Faster R-CNN. Once the network has detected the object, as shown in Figure-12a, the detection box is moved to the depth map (Figure-12b), in order to know the location of the screwdriver within the environment.



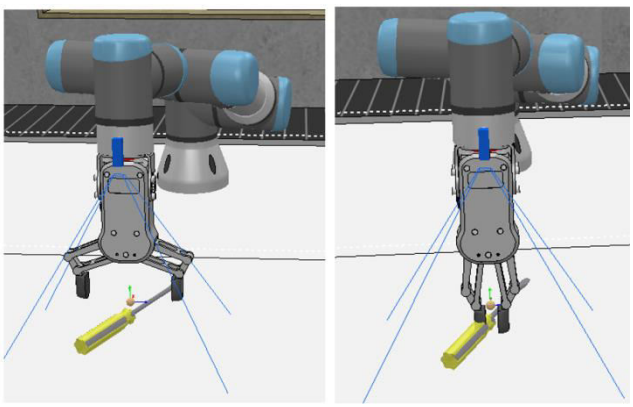

(a)




(b)

**Figure-12.** Detection of the screwdriver in (a) the photo taken by the global camera and (b) by the global depth camera.

Once the screwdriver coordinates are known, the robot is directed to the approximate location of the screwdriver by means of the kinematics function provided by the CoppeliaSim software for the UR3, as shown in Figure-13a. Once it is positioned, the end effector is closed in order to grip the object of interest, as can be seen in Figure-13b.



(a) (b)

**Figure-13.** (a) Location of the end effector on the screwdriver. (b) Grip of the object.

After the robot has picked up the object, it moves from its pickup location, passing through its initial position (Figure-14), and ending its journey on the delivery conveyor belt.

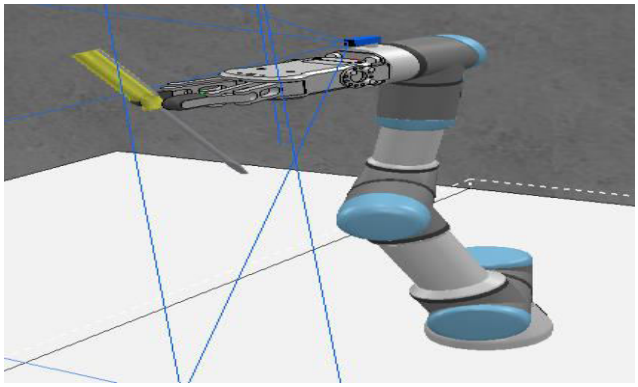
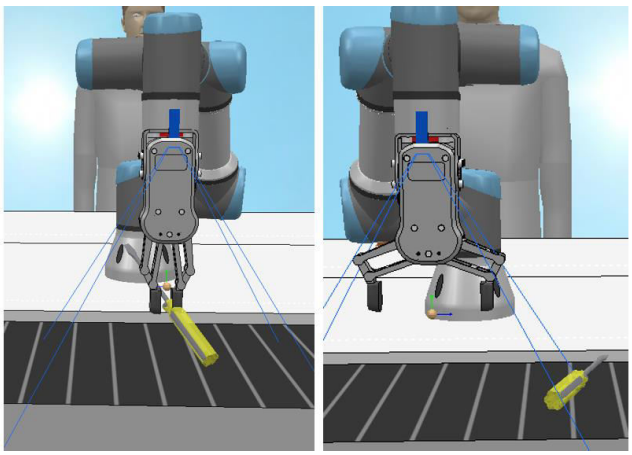


**Figure-14.** Robot passes through the start position holding the collected object.

Once located over the conveyor belt, approximately 4 cm above it, the robot opens the gripper, dropping the screwdriver on the belt, so that it is sent elsewhere, as shown in Figure-15.



(a) (b)

**Figure-15.** (a) The positioning of the robot on the conveyor belt (b) Dropping the screwdriver on the belt.

Finally, the robot returns to its initial position waiting for its next pickup.

This process is repeated 50 times to validate the virtual environment, using the Faster R-CNN, the internal kinematics system of the software and the approximate gripping capacity of the robot. From these repetitions, firstly, it is verified that the screwdriver has been located correctly. If this was not detected, the robot will not be able to perform the collection, since it will not know where the object is.

Once the detection has been validated, it is verified that the robot has grasped the object, so that when the robot starts its journey, the gripper holds the screwdriver. Likewise, it is evaluated if the robot holds the object during the path from where it collected the object to the belt, without dropping it. Finally, it is verified that when the screwdriver is dropped on the belt, it does not fall and remains inside the conveyor belt.

The results obtained are shown in Table-2. In this table it is possible to observe, for each stage of the task, the successful completion of the stage over the amounts of repetitions that were made in that stage. For example, for the stage of successful grips, there are a total of 48 repetitions done, because in the previous stage the network did not detected the tool in two trials, so only the repetitions with correct detections were evaluated in the second stage. Additionally, at the end of the table, it is shown the performance taking into account the total of repetitions done in the whole task.

**Table-2.** Results of the tests performed.

| | Correct / Repetitions | Success rate |
|---|---|---|
| Object detected correctly | 48/50 | 96% |
| Successful grips | 47/48 | 97.9% |
| Fixed grip during trajectory | 42/47 | 89.3% |
| Object Dispatch on the conveyor | 42/42 | 100% |
| Overall performance doing the whole task | 42/50 | 84% |

As it is possible to observe in the results, on only two occasions the network was not able to detect the object, because the confidence at the moment of classifying it was below the established threshold of 85%, so in those tests, the robot did not execute the task.

Within the repetitions where the network was able to locate the screwdriver, the robot could not generate the grip of the object in one try, because, when it closed the grip, it hit the object sending it to one side, but not between the two fingers, making the robot execute the task but leaving the object on the table.

When the robot was able to pick up the screwdriver, during the trajectory, on five occasions the object fell from the gripper, due to the fact that the grip made was not totally firm or, in two repetitions, the fingers of the gripper remained very close to the edge located between the handle and the blade. However, in approximately 90% of the trajectories, the screwdriver did not fall.

Finally, in all the final dispatches, from when the robot placed the screwdriver in the conveyor, they were successful, that is to say, in no delivery the object fall of the conveyor.

Overall, the entire system was able to correctly perform the entire pick and place application by 84%, even without support systems to know a proper grip or a more efficient trajectory.

## CONCLUSIONS

In this work, the construction of a virtual environment was carried out, within which various tools or objects were arranged to be used in pick and place applications, by means of a UR3 robotic device. In order to give more application scope to the environment, two deep cameras were available, so that not only the environment can be seen from a static point, but also from a dynamic one, located in the end effector.

To validate the environment, a basic pick and place application was made, using a Faster R-CNN to detect an object of interest within the environment and thus be able to execute the task of collecting and locating a screwdriver on a conveyor belt. The trained network achieved an average accuracy of 82% with the validation set, demonstrating that it was able to locate the tool in different orientations and positions within a work area. Thanks to the detection capacity of this network, within the environment validation tests, it was possible to correctly detect the object in the virtual environment 96% of the time, demonstrating the coupling capacity between the Deep Learning techniques and the simulated environments for the execution of robotic tasks.

On the other hand, although the robot was able to grab the object almost all the time, its grip was not always firm, which caused the tool to fall repeatedly. This allows us to see the possibility of applying an artificial intelligence system that gives the robot the ability to make a better decision on how to grab the tool properly, to prevent it from loosening when the robot is moving to take the object somewhere else.

## ACKNOWLEDGMENTS

The authors are grateful to the Universidad Militar Nueva Granada, which, through its Vice rectory for research, finances the present project with code IMP-ING-2935 (being in force 2019-2020) and titled “Flexible robotic prototype for feeding assistance”, from which the present work is derived.

## REFERENCES

[1] Shanshan He, Changya Yan, Yanchao Deng, Chen-Han Lee, Xiangdong Zhou. 2020. A tolerance constrained G2 continuous path smoothing and interpolation method for industrial SCARA robots. Robotics and Computer-Integrated Manufacturing. Vol. 63, 101907, ISSN 0736-5845, https://doi.org/10.1016/j.rcim.2019.101907.

[2] Hsien-I Lin. 2020. Design of an intelligent robotic precise assembly system for rapid teaching and admittance control. Robotics and Computer-Integrated Manufacturing. Vol. 64, 101946, ISSN 0736-5845, https://doi.org/10.1016/j.rcim.2020.101946.

[3] Yue Wang, Gang Zheng, Denis Efimov, Wilfrid Perruquetti. 2020. Disturbance compensation based controller for an indoor blimp robot. Robotics and Autonomous Systems. Vol. 124, 103402, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2019.103402.

[4] Qirong Tang, Zhipeng Xu, Fangchao Yu, Zhongqun Zhang, Jingtao Zhang. 2019. Dynamic target searching and tracking with swarm robots based on stigmergy mechanism. Robotics and Autonomous Systems. Vol. 120, 103251, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2019.103251.

[5] Vladimir Kvrgic, Jelena Vidakovic. 2020. Efficient method for robot forward dynamics computation. Mechanism and Machine Theory. Vol. 145, 103680,

ISSN 0094-114X, https://doi.org/10.1016/j.mechmachtheory.2019.103680.

[6] Pinzon Arenas Javier, Jimenez-Moreno Robinson, Useche Paula. 2018. Convolutional Neural Networks Training for Tools Recognition. International Journal of Applied Engineering Research ISSN: 0973-4562. 13 fasc.19: 14151- 14157.

[7] Saeed Kermani, Mostafa Ghelich Oghli, Ali Mohammadzadeh, Raheleh Kafieh. 2020. NF-RCNN: Heart localization and right ventricle wall motion abnormality detection in cardiac MRI. Physica Medica, 70: 65-74, ISSN 1120-1797, https://doi.org/10.1016/j.ejmp.2020.01.011.

[8] Vinh Quang Dinh, Farzeen Munir, Shoaib Azam, Kin-Choong Yow, Moongu Jeon. 2020. Transfer learning for vehicle detection using two cameras with different focal lengths. Information Sciences. 514: 71-87, ISSN 0020-0255. https://doi.org/10.1016/j.ins.2019.11.034.

[9] Wei Liu, Shengcai Liao, Weidong Hu. 2019. Towards accurate tiny vehicle detection in complex scenes. Neurocomputing, 347: 24-33, ISSN 0925-2312. https://doi.org/10.1016/j.neucom.2019.03.004.

[10] Yang Yu, Kailiang Zhang, Li Yang, Dongxing Zhang. 2019. Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. Computers and Electronics in Agriculture. Vol. 163, 104846, ISSN 0168-1699. https://doi.org/10.1016/j.compag.2019.06.001.

[11] Pinzon Arenas Javier, Jimenez-Moreno Robinson, Useche Paula. 2018. Faster R-CNN for Object Location in a Virtual Environment for Sorting Task. International Journal of Online and Biomedical Engineering. 14(07).

[12] Muhammad Monjurul Karim, David Doell, Ravon Lingard, Zhaozheng Yin, Ming C. Leu, Ruwen Qin. 2019. A Region-Based Deep Learning Algorithm for Detecting and Tracking Objects in Manufacturing Plants. Procedia Manufacturing. 39: 168-177, ISSN 2351-9789, https://doi.org/10.1016/j.promfg.2020.01.289.

[13] Yueyue Liu, Zhijun Li, Huaping Liu, Zhen Kan. 2020. Skill transfer learning for autonomous robots and human-robot cooperation: A survey. Robotics and Autonomous Systems. Vol. 128, 103515, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2020.103515.

[14] Qian Zhang, Guoqin Gao. 2020. Prioritizing robotic grasping of stacked fruit clusters based on stalk location in RGB-D images, Computers and Electronics in Agriculture, Vol. 172, 105359, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2020.105359.

[15] Emanuele Magrini, Federica Ferraguti, Andrea Jacopo Ronga, Fabio Pini, Alessandro De Luca, Francesco Leali. 2020. Human-robot coexistence and interaction in open industrial cells. Robotics and Computer-Integrated Manufacturing. Vol. 61, 101846, ISSN 0736-5845, https://doi.org/10.1016/j.rcim.2019.101846.

[16] Federica Ferraguti, Chiara Talignani Landi, Silvia Costi, Marcello Bonfè, Saverio Farsoni, Cristian Secchi, Cesare Fantuzzi. 2020. Safety barrier functions and multi-camera tracking for human–robot shared environment. Robotics and Autonomous Systems. Vol. 124, 103388, ISSN 0921-8890. https://doi.org/10.1016/j.robot.2019.103388.

[17] Hasan Seyyedhasani, Chen Peng, Wei-jiunn Jang, Stavros G. Vougioukas. 2020. Collaboration of human pickers and crop-transporting robots during harvesting - Part II: Simulator evaluation and robot-scheduling case-study. Computers and Electronics in Agriculture. Vol. 172, 105323, ISSN 0168-1699. https://doi.org/10.1016/j.compag.2020.105323

[18] Ren Shaoqing, *et al*. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems. pp. 91-99.

[19] He Kaiming, *et al*. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770-778.